Badih Ghazi, Neel Kamal, Ravi Kumar, Pasin Manurangsi*, and Annika Zhang

Private Aggregation of Trajectories

Abstract: In this paper, we study the task of aggregating user-generated trajectories in a differentially private manner. We present a new algorithm for this problem and demonstrate its effectiveness and practicality through detailed experiments on real-world data. We also show that under simple and natural assumptions, our algorithm has provable utility guarantees.

Keywords: trajectories, aggregation, differential privacy

DOI 10.56553/popets-2022-0125
Received 2022-02-28; revised 2022-06-15; accepted 2022-06-16.

1 Introduction

The availability of location data in mobile systems and applications has generated numerous opportunities for designing smarter services such as routing platforms, public transit, and recommendation systems. This potential, however, is limited by growing concerns for user privacy, particularly around the sharing and use of location information, reflected in an expanding set of regulations and laws [1, 22, 47].

In recent years, differential privacy (DP) [28, 29] has emerged as a popular notion for quantifying the privacy leakage of algorithms, with deployments including the 2020 US Census [3], and numerous industry use cases [6, 25, 32, 38, 53]. DP is a strong notion of privacy, ensuring that the algorithm's output does not change noticeably if the data of a single user is added or removed from the dataset (see Definition 1 for a formal statement).

Trajectories are fundamental spatio-temporal objects. They arise in a variety of settings including human mobility, animal behavior, transit, cursor/mouse movements, eye gaze patterns, etc. Aggregating trajectories is a well-studied task, often investigated along with clustering. Such aggregate information can be invaluable for several use cases such as studying different routes to go between two locations, identifying public transit routes based on trajectories, wildlife migration patterns, etc. Several prior works have considered variants of this problem, providing algorithms for clustering trajectories under different distance measures (e.g., [13, 16, 46]).

Aggregating and clustering trajectories are far more challenging problems than the corresponding versions for points in the Euclidean space. Indeed, to perform well, an algorithm has to truly exploit the underlying geometric and temporal properties of a trajectory, often encapsulated by an intricate distance measure such as Fréchet, DTW, etc. In fact, even an apparently simple one-cluster version of the problem—given a set of trajectories, find their "aggregate"—is not a straightforward task and its computational complexity depends on the underlying distance function [4, 52, 57]. In this work, we initiate the study of these problems through the lens of DP; specifically, we consider the simple task of privately aggregating trajectories.

1.1 Our Contributions

We consider the setting where each of n users is associated with a trajectory that consists of m points in the two-dimensional Euclidean space. The goal is to (privately) output a curve that well-approximates the "aggregated" trajectory. We give an efficient algorithm for this task, and prove that it satisfies the DP property. (In fact, it satisfies the strongest *user-level* DP variant.)

Furthermore, we prove that, under certain mild technical assumptions, our algorithm can recover a high-accuracy aggregate trajectory. Finally, we show the effectiveness and practicality of our algorithm by evaluating it on different real-world trajectories.

To describe the overall ideas behind our algorithm, recall that a prototypical DP algorithm works (among other things) by adding noise that is calibrated to the sensitivity of the function. It turns out that applying this idea naively to trajectory aggregation—by adding noise proportional to the radius of the input space—renders the signals in the data to be largely overwhelmed by the noise.

To overcome the aforementioned barrier, our central observation is that two consecutive points in natural trajectories are often close; their distance is *much smaller*

Badih Ghazi: Google Research, E-mail: badihghazi@gmail.com
Neel Kamal: Google, E-mail: kamalneel@google.com
Ravi Kumar: Google Research, E-mail: ravi.k53@gmail.com
***Corresponding Author: Pasin Manurangsi:** Google Research, E-mail: pasin@google.com
Annika Zhang: Google, E-mail: annikaz@google.com

than the diameter of the input space or even the length of the entire trajectory. This leads us to a notion of a *local* bounding circle, an object that allows us to aggregate each point privately while adding noise that is only roughly proportional to the distance between two consecutive points, which (as we demonstrate both experimentally and theoretically) results in a significantly better utility guarantee compared to a naive algorithm.

We note that while we focus in this work on two-dimensional aggregation, our algorithm can be naturally extended to DP aggregation of curves in higher dimensions.

1.2 Related Work

Clustering is somewhat related to track aggregation. While clustering solves the more general problem of partitioning the input trajectories into groups, we are interested in the one-cluster version: combining the input (trajectories) into a single aggregate (trajectory). In this sense, our work is closer to the 1-median setting, a special case of clustering. In the context of privacy, there has been significant recent work on DP algorithms for clustering points (see [19] and the references therein) and finding frequent patterns in sequential data [11, 12, 21]. However, none of these algorithms applies to the more challenging case of clustering trajectories. The trajectory clustering problem is usually formulated in terms of the number of clusters and some measure of the complexity of describing the centers; several clustering algorithms have been designed for different formulations [13, 16, 34, 46, 57]. Unfortunately, two reasons make these algorithms unsuitable for our problem: most of these algorithms are degenerate for the 1-median setting and to the best of our knowledge, and private versions of these algorithms have not been studied. For the latter point, we stress that, to the best of our knowledge, these algorithms cannot be made private via simple techniques. For example, [13, 16] initialize the cluster centers by (simplifications of) some input trajectories. Since each center is created from a single trajectory, making this private would require adding a vacuously large amount of noise, so large that the center is essentially a random point. A similar issue also arises in the related context of k-means/median clustering. Works on the topic of private k-means/median (e.g. [7, 19, 36, 55]) employs various sophisticated techniques to solve this issue, demonstrating its challenging nature.

A completely different approach to private track aggregation is via private synthetic data generation. The idea is to design a private generative mechanism based on the input trajectories. By the post-processing property of DP, the aggregated track can then be computed non-privately using a few synthetically generated tracks. This method was explored in a few works [35, 40]. While this method is appealing at a conceptual level, current versions of the algorithms, unfortunately, require much larger inputs, e.g., they need millions of trajectories whereas our algorithms can work with hundreds.

There has been substantial work on variants of DP tailored to the geographic setting, including [5, 20, 45], real-time traffic monitoring [33, 41], points-of-interest recommendation [42], and sharing trajectories [23]. To the best of our knowledge, none of theses prior works tackled private track aggregation.

At a technical level, the idea of "clipping" to bound the sensitivity has been employed before, e.g., for learning [2] and mean estimation [18]. However, we are not aware of any work that applies such an idea in a temporal setting similar to ours, where the local bounding circle is moved along as the algorithm progresses.

1.3 Organization

In Section 2, we recall some notation, background, and tools from previous works. We then present our algorithm in Section 3. Our theoretical and experimental results are presented in Section 4 and Section 5 respectively. We then conclude by discussing several future directions in Section 6.

2 Preliminaries

For every positive integer k, we use $[k]$ to denote $\{1, \ldots, k\}$. Unless otherwise specified, we use $\|\cdot\|$ to denote the Euclidean 2-norm. We use $B_r(c)$ to denote the two-dimensional (closed) ball of radius r centered at c, i.e., $\{x \mid \|x - c\| \leq r\}$. When c is the origin, we use B_r as a shorthand for $B_r(c)$. We denote the number of users by n.

Due to space constraints, we omit some of the proofs from the main body of the paper, and defer them to the Appendix.

2.1 Differential Privacy

We say that two datasets X, X' are *neighboring*, denoted $X \sim X'$, if X' results from adding or removing a single user from X.

Definition 1 (Differential Privacy [28, 29])**.** *For any privacy parameters $\epsilon \geq 0$ and $\delta \in [0,1]$, a randomized algorithm A is* (ϵ, δ)-differentially private (DP) *if for every pair $X \sim X'$ of neighboring datasets and for every subset S of outputs of A, it holds that $\Pr[A(X) \in S] \leq e^{\epsilon} \cdot \Pr[A(X') \in S] + \delta$, where the probabilities are over the randomness in A.*

For more background on DP, we refer the reader to the monograph [31]. Combining multiple DP algorithms results in a DP algorithm, albeit with worse parameters. This is captured next:

Lemma 2 (Basic Composition)**.** *Let $\mathcal{A}$ be an algorithm that runs $k \in \mathbb{N}$ (possibly adaptive) algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ on the same dataset[1] such that $\mathcal{A}_i$ is (ϵ_i, δ_i)-DP for $\epsilon_i, \delta_i \geq 0$ for each $i \in [k]$. Then, $\mathcal{A}$ is $(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i)$-DP.*

2.1.1 Gaussian Mechanism

Let $\mathcal{X}$ be any set, and $f : \mathcal{X}^n \to \mathbb{R}^d$ be an arbitrary d-dimensional function. Its *ℓ_2-sensitivity* is defined as $\Delta_2 f := \max_{x \sim x'} \|f(x) - f(x')\|$. The *Gaussian mechanism* with parameter σ adds independent noise sampled from $\mathcal{N}(0, \sigma^2)$ to each of the d coordinates of the output.

We let Φ denote the cumulative density function of the standard normal distribution, i.e., $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-y^2/2} dy$. The following theorem gives a tight (ϵ, δ)-DP bound for the Gaussian mechanism under adaptive compositions[2].

Theorem 3 (Theorem 5 of [54])**.** *Let $\epsilon > 0, \delta \in (0,1)$. Let $\sigma(\epsilon, \delta)$ denote the smallest value of σ such that $\delta \geq \Phi\left(-\epsilon\sigma + \frac{1}{2\sigma}\right) - e^{\epsilon}\Phi\left(-\epsilon\sigma - \frac{1}{2\sigma}\right)$. Then, applying the Gaussian mechanism with standard deviation $\Delta\sqrt{m} \cdot \sigma(\epsilon, \delta)$ on (possibly adaptive) functions $f_1, \ldots, f_m$ where $\Delta_2 f_j \leq \Delta$ for all $j \in [m]$ is (ϵ, δ)-DP.*

We note that, while this bound was stated in [54] for the special case of scalar-valued functions and non-adaptive composition, it extends readily to vector-valued functions (since the spherical symmetry of the Gaussian distribution can be used to reduce to the scalar case without loss of generality) and adaptive composition. For completeness, we provide a full proof (based on a slightly different technique of *Gaussian Differential Privacy* [26]) in Appendix A.

2.1.2 Discrete Laplace Mechanism

The discrete Laplace distribution centered at 0 and with scale $b > 0$, denoted by $\mathrm{DLap}(b)$, is the distribution with the probability mass function equal to $\frac{1}{C(b)} \cdot e^{-|k|/b}$ at every integer k, where $C(b) = \sum_{k=-\infty}^{\infty} e^{-|k|/b}$ is the normalization constant. The *discrete Laplace mechanism* with parameter b applied to an integer-valued function f adds to it noise sampled from the $\mathrm{DLap}(b)$ distribution. The *ℓ_1-sensitivity* of such a function f is defined as $\Delta f := \max_{x \sim x'} |f(x) - f(x')|$. The next lemma gives a DP bound for the discrete Laplace mechanism.

Lemma 4. *For every $\epsilon > 0$, the discrete Laplace mechanism with parameter $b \geq \Delta f / \epsilon$ is ϵ-DP.*

2.1.3 Sparse Vector Technique

We will use the so-called *sparse vector technique (SVT)* [30, 39, 51]. The input to this method is a sequence $f_1, \ldots, f_k : \mathcal{X}^n \to \mathbb{R}$ of functions together with a threshold τ and a sensitivity parameter L; its output is an index[3] $\ell^* \in [k+1]$. We say that an algorithm is (α, β)-*accurate* if its output ℓ^* satisfies the following with probability at least $1 - \beta$: $f_{\ell'}(x) < \tau$ for all $\ell' < \ell^*$ and, if $\ell^* \neq k+1$, then $f_{\ell^*}(x) \geq \tau - \alpha$. Thus the goal of an accurate algorithm is to find the first index ℓ^* at which $f_{\ell^*}(x)$ is above the given threshold. The next theorem gives an accurate algorithm for this problem.

1 By "adaptive", we mean that the choice of algorithm $\mathcal{A}_i$ can depend on the outputs of algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_{i-1}$. Here we assume that $\mathcal{A}$ does not access the dataset except through algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_k$.

2 We remark that the formulation below is slightly different compared to the exact statement of Theorem 5 of [54], because we use Φ instead of erfc and our noise is already scaled by $\Delta\sqrt{m}$.

3 Note that, for notational convenience, we assume that the method is allowed to return $k+1$ rather than $\perp$ often used in literature.

Theorem 5 (Sparse Vector Technique)**.** *Suppose that the ℓ_1-sensitivities of $f_1, \ldots, f_k$ are at most L. For every $\epsilon > 0$ and $\beta \in [0, 1/2]$, there exists an algorithm* AboveThreshold$_{L,\epsilon}$ *that is ϵ-DP and $(O(L \cdot \log(k/\beta)/\epsilon), \beta)$-accurate.*

For a more comprehensive review of SVT and its variants, see [43].

2.1.4 Partition Selection

The following primitive has been studied in prior works including [24, 37].

Definition 6 (Partition Selection)**.** *Let $\epsilon > 0$, $\delta \in (0, 1)$ be real numbers, and L and n be positive integers. In the* PartitionSelection$_{L,n,\epsilon,\delta}(T)$ *problem, each of n users gets up to L (possibly repeated) elements from a (possibly infinite) set; let T be the union of all the elements held by all users. The output is required to be (ϵ, δ)-DP, and to consist of a subset $S \subseteq T$ along with a non-negative count associated with each element in S. An algorithm for this problem is said to be* α-accurate *if for each element of S whose count is at least 2α, it is contained in the output set S and its associated count is within an additive α from its true count (i.e., its total number of occurrences in T).*

Thus the goal of an accurate algorithm is to obtain the elements that occur at least the given frequency, along with an estimate of the number of their occurrences. Note that we use user-level DP for PartitionSelection: a neighboring dataset results from adding/removing a user's ($\leq L$) points.

Theorem 7 (DP Partition Selection Algorithm)**.** *For every $\epsilon > 0$, $\delta \in (0, 0.5)$, and positive integers n and L, there is an $O(L \cdot \ln(1/\delta)/\epsilon)$-accurate (ϵ, δ)-DP algorithm for* PartitionSelection$_{L,n,\epsilon,\delta}$.

The special case of Theorem 7 where $L = 1$ has been studied in previous work, e.g., [17, 24]. Moreover, the case where $L \geq 1$ but where a user cannot hold repeated elements was studied in [10]. To the best of our knowledge, the version in Theorem 7 which is needed for our algorithms has not previously appeared in the literature. We give the proof in Appendix B. (Also note that the accuracy for PartitionSelection is a deterministic guarantee since the noise has bounded support; see Appendix B.1.)

2.2 Trajectories and Curves

A *trajectory* $\mathbf{p}$ is a sequence $(p_1, \ldots, p_m)$ of points, where $p_i = (x_i, y_i)$ and m is the *length* of the trajectory. We assume throughout that $x_i, y_i \in [-R, R]$ for some $R > 0$.

A *curve* C is a continuous mapping from $[0, 1]$ to $[-R, R]^2$.

2.2.1 Dissimilarity Metric (Fréchet distance)

We will use the Fréchet distance to measure the dissimilarity between two curves. Recall that the *Fréchet distance* between two curves C, C' is defined as

$$F(C, C') := \inf_{\alpha,\alpha'} \sup_{t\in[0,1]} \|C(\alpha(t)) - C'(\alpha'(t))\|,$$

where the infimum is over all non-decreasing and surjective functions $\alpha, \alpha' : [0, 1] \to [0, 1]$. This distance is a natural and popular curve dissimilarity measure, used in different areas, see, e.g., [8, 56] and the references therein.

Given a trajectory $\mathbf{p} = (p_1, \ldots, p_m)$ together with $0 \leq t_1 < \cdots < t_m \leq 1$, we may view $\mathbf{p}$ as a piece-wise linear curve C with the following parameterization: for any $t \in [t_1, t_m]$, write $t = \zeta t_j + (1 - \zeta)t_{j+1}$ for some $\zeta \in [0, 1]$ and $j \in [m - 1]$, and then let

$$C(t) = \zeta p_j + (1 - \zeta)p_{j+1}.$$

Furthermore, let $C(t) = p_1$ for all $t < t_1$ and $C(t) = p_m$ for all $t > t_m$. Observe that changing $t_1, \ldots, t_m$ does not change the Frechet distance between two trajectories.

By picking α such that $\alpha(\zeta j/m + (1-\zeta)(j+1)/m) = \zeta t_j + (1-\zeta)t_{j+1}$ for all $\zeta \in [0, 1], j \in [m-1]$ and similarly picking α', we arrive at the following observation:

Observation 8. *For any pair $\mathbf{p}, \mathbf{p}'$ of trajectories each of length m, we have $F(\mathbf{p}, \mathbf{p}') \leq \max_{j\in[m]} \|p_j - p'_j\|$.*

2.3 Private Trajectory Aggregation

In the *trajectory aggregation* problem, we are given n user-generated tracks $\mathbf{p}^1, \ldots, \mathbf{p}^n$. Our goal is to output an "aggregated" trajectory $\hat{\mathbf{p}}$. When $\mathbf{p}^1, \ldots, \mathbf{p}^n$ are generated from a probabilistic model with an underlying ground truth $\mathbf{p}^*$, we would like $\hat{\mathbf{p}}, \mathbf{p}^*$ to be close in the Fréchet distance. When there is no such ground truth, we simply take $\mathbf{p}^*$ to be the average $\hat{\mathbf{p}}$ of the input tracks: assuming all tracks have the same number m of points, we compute $\hat{\mathbf{p}}$ by averaging each of the m points from all tracks.

In the private version of the problem, the algorithm computing the aggregation is required to be DP. Note that we consider the stringent *user-level* DP guarantee where the output of the algorithm is required to remain indistinguishable if the entire trajectory of any user is added or removed (instead of the weaker *record-level* notion where the requirement would only apply to the addition or removal of a single point from one user's trajectory).

3 Algorithms

In this section, we present our track aggregation algorithms. Before we do so, let us try to understand why direct applications of known algorithms fail.

Perhaps the simplest way one could attempt to solve the track aggregation problem is to view each track as a $(2m)$-dimensional vector and use the Gaussian mechanism to aggregate the tracks. Unfortunately, this results in poor utility mainly because the noise required to be added to each point is proportional to the parameter R (together with a dependency on the privacy parameters). Recall that R can be very large, e.g., $[-R, R]^2$ is supposed to cover all areas of interest, such as an entire country or a continent; this results in an amount of noise that far exceeds the sum of the coordinates (for reasonable values of n and privacy parameters).

We instead propose a simple, but effective, strategy to bypass this issue via what we call a *bounding circle* (and *bounding box*). Roughly speaking, we privately identify a circle that is much smaller than $[-R, R]^2$, which contains most of the points that we would like to aggregate, and only adds noise proportional to the radius of the circle (instead of R). This is our first algorithm, named GaussianGlobalBoundingCircle, because the bounding circle is still required to "globally" cover almost all points in the tracks. Our second algorithm is a refinement of this idea, in which we instead find a "local" bounding circle that is sufficient to contain almost all points for two consecutive indices in the sequence. We then aggregate the points in each index, and move the circle along as we progress in the sequence. This allows us to add noise proportional to the radius of the local bounding circle, which in turn can be much smaller than that of the global bounding circle.

3.1 Bounding Circle Computation

Our algorithm employs what we call a *bounding box* and a *bounding circle*. The former is simply a square box $[x_{\text{lo}}, x_{\text{up}}) \times [y_{\text{lo}}, y_{\text{up}})$ while the latter is a circle of radius r centered at a point (x, y).

We provide an algorithm, denoted by BoundingCircle (Figure 1), that, starting with the trivial bounding box $[-R, R]^2$, refines it into a smaller bounding circle that can be later used in aggregation. Our algorithm is divided into two phases: in the first phase, we use the sparse vector technique (Theorem 5) to find the box side length r. To compute each $f_i(S)$, we divide $[-R, R]^2$ into grid cells of side length $r = R/2^{k+1-i}$ and let $f_i(S)$ denote the largest number of points that belong to the same cell. Notice that $f_i(S)$ is increasing in i and, that if $f_i(S)$ exceeds the threshold of τ, then there is a "good" bounding box of length $r = R/2^{k+1-i}$ that contains τ points. The latter is not yet sufficient to provide a sufficient accuracy guarantee because it is possible that there is in fact a bounding box of length r that contains say 1.1τ points but our division cuts this cell into four pieces where each piece contains $1.1\tau/4 < 0.3\tau$ points. This issue however can be solved relatively easily by taking a "random shift" (denoted by s_x, s_y in the algorithm) before dividing the initial region into grid cells.

The full pseudo-code of our algorithm is presented in Figure 1. We remark that for the purpose of our experiments, we implement AboveThreshold as in [31, Algorithm 1] and implement PartitionSelection as in [24, 37]. Since the primitives used in our algorithm are just the sparse vector technique and the partition selection algorithm, it is simple to argue its privacy guarantee:

Lemma 9. *Algorithm* BoundingCircle *is* (ϵ_b, δ_b)*-DP.*

Proof of Lemma 9. First, observe that the sensitivity of each of $f_1, \dots f_k$ is at most L. Indeed, the input consists of L points per user and hence adding/removing a user can change f_i by at most L. Thus, by Theorem 5, the index i^* computed in line 10 of Algorithm 1 is $\epsilon_b^{\text{radius}}$-DP. Moreover, the output $\{\widehat{\text{count}}_G : G \in \mathcal{G}_r\}$ of the call on line 17 to the $\text{PartitionSelection}_{L,n,\epsilon,\delta}$ procedure is $(\epsilon_b^{\text{box}}, \delta_b)$-DP by Theorem 7. By basic composition (Lemma 2), the output c, r of Algorithm 1 is $(\epsilon_b^{\text{radius}} + \epsilon_b^{\text{box}}, \delta_b)$-DP. The lemma now follows from the setting of $\epsilon_b^{\text{radius}}$, ϵ_b^{box} in line 3 of Algorithm 1. □

We remark that other algorithms for similar tasks (sometimes referred to as *1-Cluster* in literature) have

Fig. 1. Algorithm for bounding circle.

Algorithm BoundingCircle.

1: **Input:** A multiset S of points, threshold $\tau \in \mathbb{N}$ of desired number of points in the circle.
2: **Parameters:** Privacy parameters $\epsilon_b, \delta_b > 0$, maximum number $L \in \mathbb{N}$ of points from each user, maximum depth $k \in \mathbb{N}$ of radius search.
3: $\epsilon_b^{\text{radius}}, \epsilon_b^{\text{box}} \leftarrow \epsilon_b/2$
{Phase I: Find Radius}
4: $s_x, s_y \leftarrow$ uniform random number $\in [-R, 0]$
{Random Shifts}
5: **for** $\ell = 1, \dots, k$ **do**
6: $r \leftarrow R/2^{k+1-\ell}$
7: $\mathcal{G}_r \leftarrow \{[x, x+r) \times [y, y+r) \mid x \in \{-R+s_x, -R+r+s_x, \dots, 2R+s_x\}$ and $y \in \{-R+s_y, -R+r+s_y, \dots, 2R+s_y\}\}$
8: $f_\ell(S) \leftarrow \max_{G \in \mathcal{G}_r} |S \cap G|$
9: **end for**
10: $\ell^* \leftarrow \text{AboveThreshold}_{L, \epsilon_b^{\text{radius}}}(f_1, \dots, f_k, \tau)$
11: $r \leftarrow R/2^{k+1-\ell^*}$
{Phase II: Find Box}
12: $T \leftarrow$ empty multiset.
13: **for** all $i \in S$ **do**
14: $G \leftarrow$ interval in $\mathcal{G}_r$ that contains i
15: Add G to multiset T
16: **end for**
17: $(\widehat{\text{count}}_G)_{G \in T} \leftarrow \text{PartitionSelection}_{L, n, \epsilon_b^{\text{box}}, \delta_b}(T)$
18: $[x_{\text{lo}}, x_{\text{up}}) \times [y_{\text{lo}}, y_{\text{up}}) \leftarrow \arg\max_{G \in T} \widehat{\text{count}}_G$
19: $c = ((x_{\text{lo}} + x_{\text{up}})/2, (y_{\text{lo}} + y_{\text{up}})/2)$
20: **return** c, r

been devised in [36, 48–50], but all of these works mainly focus on theoretical guarantees for high dimensions and thus their algorithms are much more involved. Our algorithm is simple and still yields a satisfactory guarantee for our purposes.

3.2 Baseline: Global Bounding Circle

Our first algorithm is one that simply computes a bounding circle for all the input points. Then, for each index $j \in [m]$, it computes the aggregate by projecting the points onto the ball and aggregating them using the Gaussian mechanism. For the latter, note that in order to keep the sensitivity small we have to subtract the center from each projected point before adding it back to the aggregate. Figure 2 contains a formal description. It deviates slightly from the previous simplified description in that it also contains an "enlargement factor" λ that gets multiplied to the radius r (Line 8). This helps us avoid considering circles that are too small, which would prevent us from capturing the points outside the bounding circle. Indeed, if a bounding circle is too small, then many points might be outside it, resulting in error as points are projected to it before averaging. On the other hand, if the bounding circle is too large, then we have to add a larger noise, also resulting in more error. From our experience, the former error is larger, hence we enlarge the circle.

In Line 6 of the algorithm, the constant 0.6 is an arbitrary choice; any value more than 0.5 works and will ensure that the circle contains both the first points and second points from the tracks. In Line 12, it is necessary to divide by $\hat{n}$ to hide the the number of points, since the DP notion we use is under addition/removal of users.

Fig. 2. Algorithm for global bounding circle.

Algorithm GaussianGlobalBoundingCircle.

1: **Input:** Trajectories $\mathbf{p}^1, \dots, \mathbf{p}^n$ each containing m points.
2: **Parameters:** Privacy parameters $\epsilon_a, \delta_a, \epsilon_b, \delta_b, \epsilon_c > 0$, ratio $\lambda > 1$ used to inflate the bounding circle, maximum depth $k \in \mathbb{N}$ of radius search in BoundingCircle.
3: $\hat{n} \leftarrow n + \text{DLap}(1/\epsilon_c)$
4: $S \leftarrow$ multiset of all points from $\mathbf{p}^1, \dots, \mathbf{p}^n$
5: $L \leftarrow m$
6: $\tau \leftarrow 0.6\hat{n}m$
7: $c, r \leftarrow \text{BoundingCircle}_{\epsilon_b, \delta_b, L, k}(S, \tau)$
8: $r' \leftarrow \lambda \cdot r$
9: $\sigma' \leftarrow \sqrt{m} r' \cdot \sigma(\epsilon_a, \delta_a)$ {From Theorem 3}
10: $\Pi \leftarrow$ projection operator from $\mathbb{R}^2$ to $B_{r'}$, i.e., $\Pi(x)$ is the point in $B_{r'}$ closest to x
11: **for** $j \in [m]$ **do**
12: $\hat{p}'_j \leftarrow \frac{1}{\hat{n}} \left(\left(\sum_{i \in [n]} \Pi(p_j^i - c) \right) + \mathcal{N}(0, \sigma'^2)^{\otimes 2} \right)$
13: $\hat{p}_j \leftarrow c + \Pi(\hat{p}'_j)$
14: **end for**
15: **return** $\hat{\mathbf{p}} = (\hat{p}_1, \dots, \hat{p}_m)$

Due to the projection and subtraction, it is easily seen that the ℓ_2-sensitivity of each point in the aggregation is at most r'. This allows us to argue the privacy of the algorithm as follows.

Lemma 10. *Algorithm* GaussianGlobalBoundingCircle *is* $(\epsilon_a + \epsilon_b + \epsilon_c, \delta_a + \delta_b)$*-DP.*

Proof of Lemma 10. By the DP property of the discrete Laplace mechanism (Lemma 4), the value $\hat{n}$ computed on line 3 of Figure 2 is ϵ_c-DP. Using the DP property of the BoundingCircle algorithm (Lemma 9) along with basic composition (Lemma 2), we get that the center c and radius r computed on Line 7 of Figure 2 are $(\epsilon_b + \epsilon_c, \delta_b)$-DP. Finally, let $\mathcal{X} = \mathbb{R}^2$, and consider the function $f : \mathcal{X}^n \to \mathbb{R}^2$ given by $f(x) := \sum_{i\in[n]} \Pi_{r'}(x_i - c)$ (where c is fixed, as computed in line 7). Note that the ℓ_2-sensitivity of f satisfies $\Delta_2(f) \le r'$. Thus, using the DP property of the Gaussian mechanism (Lemma 3) along with another application of basic composition (Lemma 2), we conclude that the output $\hat{\mathbf{p}}$ of Figure 2 is $(\epsilon_a + \epsilon_b + \epsilon_c, \delta_a + \delta_b)$-DP. □

3.3 Improvement: Local Bounding Circle

Our improved algorithm, denoted by GaussianLocalBoundingCircle, instead finds the bounding circle using just the first two points from each track. This bounding circle is then used to aggregate the first points of the track. We then move the center of the bounding circle to this aggregated point. This process is repeated for the ith point in the tracks for $i = 2, \ldots, n$, where the center used is the result from the previous (i.e., $(i-1)$th) aggregation. A demonstration of the algorithm is shown in Figure 4.

The privacy analysis of this algorithm is similar to its global bounding circle counterpart.

Lemma 11. *Algorithm* GaussianLocalBoundingCircle *is* $(\epsilon_a + \epsilon_b + \epsilon_c, \delta_a + \delta_b)$*-DP.*

We end by noting that Figure 3 runs in time $O(mn+kn)$, because finding the bounding circle (Line 7) takes $O(kn)$ time whereas the remaining steps require $O(mn)$ time. Note that we typically set $k < m$; in this regime of parameters, the running time is $O(mn)$, which is linear in the input size.

Proof of Lemma 11. By the DP property of the discrete Laplace mechanism (Lemma 4), the value $\hat{n}$ computed on line 3 of Figure 3 is ϵ_c-DP. Using the DP property of the BoundingCircle algorithm (Lemma 9) along with basic composition (Lemma 2), we get that the center c_0 and radius r computed on line 7 of Figure 3 are $(\epsilon_b + \epsilon_c, \delta_b)$-DP. Finally, let $\mathcal{X} = \mathbb{R}^2$, and consider the functions $f_j : \mathcal{X}^n \to \mathbb{R}^2$, for all $j \in [m]$, given by $f_j(x) := \sum_{i\in[n]} \Pi_{r'}(x_i - c_{j-1})$ (where c_j's are set on lines 7 and 14). Note that c_{j-1} depends on the outputs of $f_1, \ldots, f_{j-1}$, and that the ℓ_2-sensitivity of each f_j satisfies $\Delta_2(f_j) \le r'$. Thus, using the DP property of the Gaussian mechanism (Theorem 3 applied to the composition of m adaptive queries) along with another application of basic composition (Lemma 2), we conclude that the output $\hat{\mathbf{p}}$ of Figure 3 is $(\epsilon_a + \epsilon_b + \epsilon_c, \delta_a + \delta_b)$-DP. □

Fig. 3. Algorithm for local bounding circle.

Algorithm GaussianLocalBoundingCircle.

1: **Input:** Trajectories $\mathbf{p}^1, \ldots, \mathbf{p}^n$ each containing m points.
2: **Parameters:** Privacy Parameters $\epsilon_a, \delta_a, \epsilon_b, \delta_b, \epsilon_c > 0$, the ratio $\lambda > 1$ used to inflate the bounding circle, maximum depth $k \in \mathbb{N}$ of radius search in BoundingCircle.
3: $\hat{n} \leftarrow n + \mathrm{DLap}(1/\epsilon_c)$
4: $S \leftarrow \{p_1^1, p_2^1, \ldots, p_1^n, p_2^n\}$ {First two points}
5: $L \leftarrow 2$
6: $\tau \leftarrow 1.2\hat{n}$
7: $c_0, r \leftarrow \text{BoundingCircle}_{\epsilon_b,\delta_b,L,k}(S, \tau)$
8: $r' \leftarrow \lambda \cdot r$
9: $\sigma' \leftarrow \sqrt{m} r' \cdot \sigma(\epsilon_a, \delta_a)$ {From Theorem 3}
10: $\Pi \leftarrow$ projection operator from $\mathbb{R}^2$ to $B_{r'}$
11: **for** $j \in [m]$ **do**
12: $\quad \hat{p}'_j \leftarrow \frac{1}{\hat{n}} \left(\left(\sum_{i\in[n]} \Pi(p_j^i - c_{j-1}) \right) + \mathcal{N}(0, \sigma'^2)^{\otimes 2} \right)$
13: $\quad \hat{p}_j \leftarrow c_{j-1} + \Pi(\hat{p}'_j)$
14: $\quad c_j \leftarrow \hat{p}_j$
15: **end for**
16: **return** $\hat{\mathbf{p}} = (\hat{p}_1, \ldots, \hat{p}_m)$

4 Utility Analysis

In this section, we prove that our GaussianLocalBoundingCircle algorithm outputs a provably good aggregate trajectory, under a few mild assumptions. The track aggregation problem has typically been formulated as an optimization problem, in which the goal is to minimize a certain (1-mean or 1-median) objective function (e.g., [14, 15, 27]). Unfortunately, these algorithms, which have some theoretical guarantees, are far more complicated than ours, and it is unclear how to make them provably DP. On the other hand, there are several works that provide algorithms for track aggregation, but with no performance guarantees (e.g., [16]).

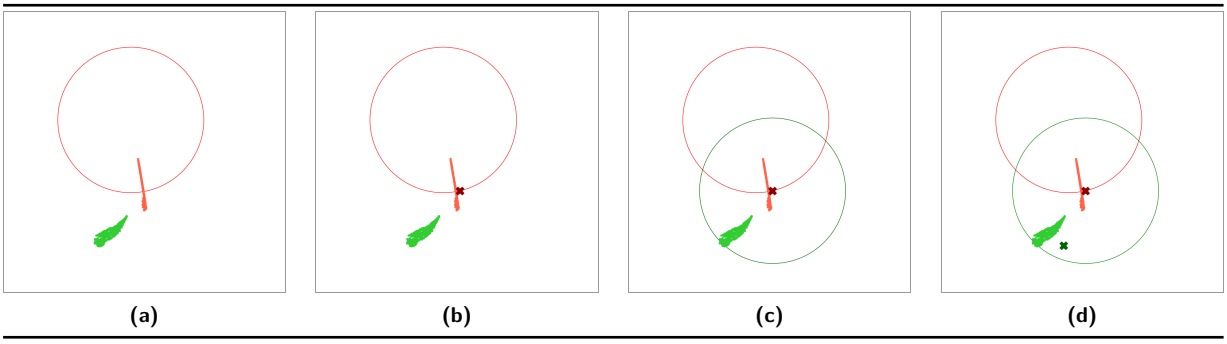

Fig. 4. An illustration of GaussianLocalBoundingCircle algorithm at stage i of the algorithm. Red objects correspond to the ith stage and green to the $(i+1)$th stage. For a given color, the cloud of points are from the input trajectories, the one marked $\times$ is the privately aggregated point, and the circle is the bounding circle. In (b), the ith privately aggregated point is first obtained by a noisy average of the ith points from the trajectories projected onto the ith bounding circle. In (c), the $(i+1)$th bounding circle is centered at the ith aggregated point and has the same radius as the ith circle.

4.1 Model and Assumptions

To quantify the performance of our algorithms, we follow a modeling-based approach: we assume that there is an underlying curve from which each input trajectory is observed, and our objective is to minimize the (Fréchet) distance between the output (aggregated) trajectory and the underlying curve.

Even in this setting, it is still not yet possible to prove any non-trivial bound. For example, if all the points in the trajectories are just the same point, then there is no way to reconstruct other points on the curve. As such, we need additional assumptions to allow us to prove concrete guarantees. In addition to the assumption that every point comes from the underlying curve that we call *accurate samples*, roughly speaking, there are three properties we assume:

- *Lipschitzness*: no two consecutive points are too far apart.
- *Initial Inverse-Lipschitzness*: the first two points in each trajectory are sufficiently far apart.
- *Cross-User Synchronization*: the jth points of all trajectories come before the $(j+1)$th points of all trajectories.

The Lipschitzness property is perhaps the most intuitive: every data-collection device (e.g., GPS) that records at regular time intervals has a physical limitation on how far the object can move during that interval. Initial inverse-Lipschitzness generally says that the user cannot stay idle at the starting point; it is typically not hard to enforce, e.g., by removing the first few points that are too close to the starting point. Cross-user synchronization is perhaps the hardest to motivate but it can be reasonable in the case where the object moves at roughly the same rate along the curve (e.g., trains or scheduled buses); in other cases, one might attempt to achieve it by interpolating and resampling each trajectory at the same rate. (Pre-processing does not affect DP guarantees when it is applied on each input individually.)

We formalize these properties in Assumption 12. Before we move forward, we remark that our guarantees still hold for relaxed versions of these assumptions. We defer this discussion to Section 4.4.

Assumption 12. *There exists a curve* $C : [0,1] \to [-R,R]^2$ *and an number* $r_{\mathrm{lo}}, r_{\mathrm{up}} \in [\gamma R, R]$ *where* $\gamma \in (0,1]$ *such that:*

- *(Accurate Samples) For every* $i \in [n], j \in [m]$*, there exists a* $t_j^i \in [0,1]$ *such that* $p_j^i = C(t_j^i)$*.*
- *(Cross-User Synchronization) Let* $t_j^{\max} := \max_{i\in[n]} t_j^i$ *and* $t_j^{\min} := \min_{i\in[n]} t_j^i$*. Then, for all* $j \in [m-1]$*,* $t_{j+1}^{\min} > t_j^{\max}$*.*
- *(Lipschitzness) For every* $j \in \{0,\dots,m\}$ *and* $t, t' \in [t_j^{\min}, t_{j+1}^{\max}]$*, we have* $\|C(t) - C(t')\| < r_{\mathrm{up}}$ *where we use the convention* $t_0^{\min} = t_0^{\max} = 0$ *and* $t_{m+1}^{\min} = t_{m+1}^{\max} = 1$*.*
- *(Initial Inverse-Lipschitzness) For every* $i \in [n]$*,* $\|C(t_1^i) - C(t_2^i)\| \geq r_{\mathrm{lo}}$*.*

We stress that Assumption 12 is made only for utility analysis and *not* for privacy, which holds even for worst-case datasets. Also note that there is no obvious way to check if real-world datasets satisfy Assumption 12.

Given Assumption 12, we can establish the following bound on the Fréchet distance between the underlying curve and the DP aggregated trajectory output by GaussianLocalBoundingCircle.

Corollary 13. *Let $\epsilon_a = \epsilon_b = \epsilon_c = \epsilon/3$ and $\delta_a = \delta_b = \delta/2$. Suppose that Assumption 12 holds with $\gamma \geq 2^{-k}$ and $r_{\mathrm{up}}/r_{\mathrm{lo}} = O(1)$. If $n \geq \kappa(\log(k/\delta) + \sqrt{m \log m \log(1/\delta)})/\epsilon$ for some sufficiently larger constant κ and if the parameter λ of the algorithm is a sufficiently large constant, then with probability at least 3/4, the output $\hat{\mathbf{p}}$ of* GaussianLocalBoundingCircle *satisfies $F(C, \hat{\mathbf{p}}) \leq O(r_{\mathrm{up}})$.*

We remark that our bound is at least $\Omega(r_{\mathrm{up}})$ regardless of the value of n. This might be surprising at first glance, but it is actually to be expected: even under Assumption 12, it is still possible that all trajectories are the same and every pair of consecutive points are r_{up} apart. In this case, there is no way to reconstruct the points on the curve between these m points, and thus a Fréchet distance of $\Omega(r_{\mathrm{up}})$ is necessary. Note that if we were to use the global bounding circle, the error bound would scale linearly with the radius of the smallest ball that contains all points, which may be as large as $m \times r_{\mathrm{up}}$. (Similarly, the trivial bounding box would lead to a bound that scales linearly with R.)

4.1.1 Concrete Parameters and Bounds

Although for simplicity of presentation we did not compute the exact constants in our theoretical analysis (e.g., κ in Corollary 13), we remark that our bounds can still give fairly concrete guarantees. For example, suppose that $k = 20$, which allows our assumption on $r_{\mathrm{lo}}, r_{\mathrm{up}}$ (in Assumption 12) to be $[2^{-20}R, R]$. If we take the upper bound R to be generous, e.g., the diameter of Earth, $2^{-20}R$ is still less than 15 meters; this range is likely to be sufficient for any real-world geographic applications. Similarly, suppose that we have as many as $m = 1000$ points per trajectory. Furthermore, suppose that we are selecting the privacy parameters ϵ, δ to be 1 and 10^{-8} respectively. Then, the quantity $(\log(k/\delta) + \sqrt{m \log m \log(1/\delta)})/\epsilon$ in Corollary 13 is smaller than 165. Even though we did not calculate κ precisely, it is not hard to verify that it is no more than 20 when $r_{\mathrm{up}}/r_{\mathrm{lo}} = 2$. In this case, we therefore only require $n \geq 3300$ trajectories for our theoretical guarantees to hold.

4.1.2 Proof Overview

In fact, Corollary 13 is a simplified version of a more general guarantee from Theorem 18, which we state in Section 4.3 below. We now briefly describe the high-level ideas of the proof. First, we show that the bounding circle we find is "good". Roughly speaking, this means that the starting circle c_0, r should contain all the first points in the trajectories, and that the value of r is in the "right" range. The latter is important because if r were too large, we would be adding too much noise; on the other hand, if r were too small, the bounding circles may not contain subsequent points in aggregation, leading to large errors. Indeed, we will show, using the guarantee of AboveThreshold, that r is between $\Omega(r_{\mathrm{lo}})$ and $O(r_{\mathrm{up}})$, meaning that if we set λ sufficiently large, we will have $r \gg r_{\mathrm{up}}$. The latter, together with concentration bounds for the (discrete) Laplace and Gaussian noise distributions, will allow us to argue that all the points we aggregate in each step always belong to the bounding circle. This means that the error between the (unnoised) average and the output $\hat{\mathbf{p}}$ only comes from the Gaussian noise, which we can bound. Furthermore, we show below that, when Assumption 12 holds, the (unnoised) average is a good approximation of the underlying curve (Lemma 15). Combining these two bounds gives us our main utility guarantee.

4.1.3 Additional Notation and Conventions

For every $j \in [m]$, let $\tilde{p}_j = \frac{1}{n}\left(\sum_{i \in [n]} \Pi_j(p_j^i)\right)$ and let $\tilde{\mathbf{p}} = (\tilde{p}_1, \ldots, \tilde{p}_m)$ denote the (unnoised) average trajectory. In this section, we will assume that κ is a sufficiently large constant and that β is any real value in $(0, 0.5)$; these will not be made explicit in the formal statements for brevity.

The following are simple useful observations that can be obtained from Assumption 12. The first is an upper bound on the distance between two nearby points, using the Lipschitzness property.

Observation 14. *If Assumption 12 holds, then for every $i, i' \in [n]$ and $j, j' \in [m]$ where $|j - j'| \leq 1$, we have $\|p_j^i - p_{j'}^{i'}\| \leq r_{\mathrm{up}}$.*

Proof of Observation 14. We may assume without loss of generality that $j \leq j'$. Recall that $p_j^i = C(t_j^i), p_{j'}^{i'} = C(t_{j'}^{i'})$. Appealing to Cross-User Synchronization, we have $t_j^i, t_{j'}^{i'} \in [t_j^{\min}, t_{j'}^{\max}]$. Applying the Lipschitzness property then yields the desired bound. □

The second is an upper bound on the Fréchet distance between the underlying curve and the unnoised average trajectory, obtained by a careful case analysis.

Lemma 15. *If Assumption 12 holds, then we have* $F(C, \tilde{\mathbf{p}}) \le r_{\mathrm{up}}$.

Proof of Lemma 15. Let us view $\tilde{\mathbf{p}} = (\tilde{p}_1, \dots, \tilde{p}_m)$ as a curve C' with the following parameterization (see Section 2.2.1): $t_j = t_j^{\min}$ for all $j \in [m]$. More specifically, for any $t \in [t_1^{\min}, t_m^{\min})$, write $t = \zeta t_j^{\min} + (1-\zeta) t_{j+1}^{\min}$ for some $\zeta \in [0,1)$ and $j \in [m+1]$, and then let

$$C'(t) = \zeta \tilde{p}_j + (1-\zeta)\tilde{p}_{j+1}.$$

Furthermore, for $t \in [0, t_1^{\min})$ and $t \in [t_m^{\min}, 1)$, we let $C(t) = \tilde{p}_1$ and $C(t) = \tilde{p}_m$ respectively.

Notice that we have

$$F(C, \tilde{\mathbf{p}}) = F(C, C') \le \max_{t \in [0,1]} \|C(t) - C'(t)\|_2.$$

Thus, it suffices to show that $\|C(t) - C'(t)\|_2 \le r_{\mathrm{up}}$ for all $t \in [0,1]$. To prove this, consider three cases, based on whether $t < t_1^{\min}$, $t \in [t_1^{\min}, t_m^{\min})$ or $t \ge t_m^{\min}$.

Case I: $t < t_1^{\min}$**.** We have

$$\begin{aligned}
\|C(t) - C'(t)\|_2 &= \|C(t) - \tilde{p}_1\|_2 \\
&= \left\| C(t) - \frac{1}{n}\sum_{i \in [n]} p_1^i \right\|_2 \\
&\le \frac{1}{n}\sum_{i \in [n]} \|C(t) - p_1^i\|_2 \\
&= \frac{1}{n}\sum_{i \in [n]} \|C(t) - C(t_1^i)\|_2.
\end{aligned}$$

Notice that each of $t, t_1^1, \dots, t_1^n$ belongs to $[0, t_1^{\max}] = [t_0^{\min}, t_1^{\max}]$. Applying the Lipschitzness condition from Assumption 12, we can conclude $\|C(t) - C'(t)\|_2 \le r_{\mathrm{up}}$.

Case II: $t \ge t_m^{\min}$**.** Similar to the previous case, we have $\|C(t) - C'(t)\|_2 \le \frac{1}{n}\sum_{i \in [n]} \|C(t) - C(t_m^i)\|_2 \le r_{\mathrm{up}}$ where the latter inequality follows from the Lipschitzness condition from Assumption 12 together with the fact that $t, t_m^1, \dots, t_m^n \in [t_m^{\min}, 1] = [t_m^{\min}, t_{m+1}^{\max}]$.

Case III: $t \in [t_1^{\min}, t_m^{\min})$**.** In this case, $t = \zeta t_j^{\min} + (1-\zeta) t_{j+1}^{\min}$ for some $\zeta \in [0,1)$ and $j \in [m]$. We may bound $\|C(t) - C'(t)\|_2$ as follows:

$$\begin{aligned}
&\|C(t) - C'(t)\|_2 \\
&= \|C(t) - \zeta\tilde{p}_j - (1-\zeta)\tilde{p}_{j+1}\|_2 \\
&\le \zeta \cdot \|C(t) - \tilde{p}_j\|_2 + (1-\zeta)\cdot\|C(t) - \tilde{p}_{j+1}\|_2 \\
&= \zeta \cdot \left\| C(t) - \frac{1}{n}\sum_{i \in [n]} p_j^i \right\|_2 \\
&\quad + (1-\zeta)\cdot \left\| C(t) - \frac{1}{n}\sum_{i \in [n]} p_{j+1}^i \right\|_2 \\
&\le \frac{\zeta}{n}\sum_{i \in [n]} \left\| C(t) - p_j^i \right\|_2 + \frac{1-\zeta}{n}\sum_{i \in [n]} \left\| C(t) - p_{j+1}^i \right\|_2 \\
&= \frac{\zeta}{n}\sum_{i \in [n]} \left\| C(t) - C(t_j^i) \right\|_2 \\
&\quad + \frac{1-\zeta}{n}\sum_{i \in [n]} \left\| C(t) - C(t_{j+1}^i) \right\|_2.
\end{aligned}$$

Finally, observe that $t, t_j^1, \dots, t_j^n, t_{j+1}^1, \dots, t_{j+1}^n \in [t_j^{\min}, t_{j+1}^{\max}]$. Thus, applying the Lipschitzness condition from Assumption 12, we can conclude that $\|C(t) - C'(t)\|_2 \le r_{\mathrm{up}}$ in this case as well. □

4.2 Bounding Circle Guarantee

To prove that the bounding circle we compute is "good", we start by proving the guarantee on the radius r that is found:

Lemma 16. *Suppose that Assumption 12 holds with* $\gamma \ge 2^{-k}$*. If* $n \ge \kappa(\log(1/\beta)/\epsilon_c + \log(k/\beta)/\epsilon_b)$*, then with probability* $1 - 5\beta$*, the value* r *computed on line 11 of Figure 1 satisfies* $r \in [r_{\mathrm{lo}}/\sqrt{2}, 2r_{\mathrm{up}}/\beta]$.

Proof of Lemma 16. First, with probability $1-\beta$, we have that $|n - \hat{n}| \le \Delta_n = O(\log(1/\beta)/\epsilon_c)$. Conditioned on this event, we will separately prove that $r \ge r_{\mathrm{lo}}/\sqrt{2}$ holds with probability $1-\beta$ and that $r \le 2r_{\mathrm{up}}/\beta$ holds with probability $1-3\beta$. A union bound will yield the desired claim.

First, let us show that $r \ge r_{\mathrm{lo}}/\sqrt{2}$ with probability $1-\beta$. Let ℓ^* be the index returned on line 10. Notice that if $\ell^* = k+1$, then we simply have that $r = R$ and thus the condition is trivially satisfied. Otherwise, if $i^* \le k$, then the guarantee of the AboveThreshold algorithm (Theorem 5) ensures that, with probability $1-\beta$, we have $f_{\ell^*}(S) \ge \tau - \alpha = 1.2\hat{n} - O(\log(k/\beta)/\epsilon_b) \ge 1.2n - \Delta_n - O(\log(k/\beta)/\epsilon_b)$. When $n \ge \kappa \cdot \log(k/\beta)/\epsilon_b$ for a sufficiently large constant κ, this lower bound is larger than n. This means that for some $i \in [n]$, both $p_1^i = C(t_1^i)$ and $p_2^i = C(t_2^i)$ belong to some cell $G \in \mathcal{G}_r$. This implies that $\|p_1^i - p_2^i\| \le \sqrt{2}\cdot r$. However, the inverse Lipschitzness property in Assumption 12 asserts that the left hand side quantity must be at least r_{lo}. As such, we can conclude that $r \ge r_{\mathrm{lo}}/\sqrt{2}$ as desired.

Next, we show that $r \le 2r_{\mathrm{up}}/\beta$ with probability $1-3\beta$. If $r_{\mathrm{up}} \ge \beta R/2$, then this obviously holds; thus, we may assume henceforth that $r_{\mathrm{up}} < \beta R/2$. Observation 14 implies that $p_1^1, p_2^1, \dots, p_1^n, p_2^n$ all lie within some $r_{\mathrm{up}} \times r_{\mathrm{up}}$ box B. For $r^0 \ge r_{\mathrm{up}}$, observe that the proba-

bility (over s_x, s_y) that B is a subset of some $G \in \mathcal{G}_{r^0}$ is at least $1 - 2r_{\text{up}}/r^0$.

Let $\hat{\ell} := k+1-\lfloor \log_2(\beta R/r_{\text{up}}) \rfloor$ and $\hat{r} = R/2^{k+1-\hat{\ell}} \geq r_{\text{up}}/\beta$. From the last bound derived in the previous paragraph, B is contained in some $G \in \mathcal{G}_{\hat{r}}$ with probability $1-2\beta$. When this occurs, we have $f_{\hat{\ell}}(S) \geq n$. Moreover, the guarantee of the AboveThreshold algorithm (Theorem 5) ensures that, with probability $1-\beta$, $f_\ell(S) < \tau < n$ for all $\ell < \ell^*$. This means that $\ell^* \geq \hat{\ell}$. Since we eventually set $r = 2^{k+1-\ell^*} \geq 2^{k+1-\hat{\ell}}$, we have $r \leq 2r_{\text{up}}/\beta$ with probability at least $1-3\beta$ as desired. □

Having argued that r is in the desired range, we can now show that the bounding circle itself satisfies two properties that will allow the analysis of the noise addition step to go through.

Lemma 17. *Suppose that Assumption 12 holds with $\gamma \geq 2^{-k}$. If $n \geq \kappa(\log(1/\beta)/\epsilon_c + \log(k/\beta)/\epsilon_b + (r_{\text{up}}/r_{\text{lo}})^2 \cdot \log(1/\delta_b)/\epsilon_b)$ and $\lambda \geq 3(r_{\text{up}}/r_{\text{lo}})/\beta$, then with probability $1-5\beta$, then center c_0 and radius r computed on Line 7 of Algorithm 3 satisfies the following:*

1. *$p_1^1, \ldots, p_1^n \in B_{\lambda r}(c_0)$, and,*
2. *For every $i \in [n]$ and $j \in [m-1]$, $\|p_{j+1}^i - \tilde{p}_j\| \leq 0.5\lambda r$.*

Proof of Lemma 17. From Lemma 16, we have $r \in [r_{\text{lo}}/\sqrt{2}, 2r_{\text{up}}/\beta]$ with probability $1-5\beta$. Recall from the proof of Lemma 16 that $p_1^1, p_2^1, \ldots, p_1^n, p_2^n$ belong to some $r_{\text{up}} \times r_{\text{up}}$ box B. Hence, there exists $G \in \mathcal{G}_r$ s.t.

$$\begin{aligned} |S \cap G| &\geq 2n(r/r_{\text{up}})^2/4 \\ (\text{From } r \geq r_{\text{lo}}/\sqrt{2}) &\geq n(r_{\text{lo}}/r_{\text{up}})^2/4 \\ &\geq 0.25\kappa \cdot \log(1/\delta_b)/\epsilon_b. \end{aligned}$$

When κ is sufficiently large, Theorem 7 implies that the error incurred in each reported count is less than $|S \cap G|/2$. This means that the selected box must contain at least one point $p_j^i \in S$ (where $i \in [n]$ and $j \in \{1,2\}$). Conditioned on this, we have $\|c - p_j^i\| \leq r/\sqrt{2} \leq \sqrt{2}r_{\text{up}}/\beta$, which, from the fact that all points in S belong to an $r_{\text{up}} \times r_{\text{up}}$ box B, implies that $\|c - p\| \leq (\sqrt{2}/\beta + 1)r_{\text{up}}$. From our choice of λ, this is no more than $\lambda \cdot r$. Hence, we have that $p_1^1, \ldots, p_1^n \in B_{\lambda r}(c_0)$ as desired.

We can now prove the second property. By the triangle inequality and Observation 14 respectively, we have

$$\|p_{j+1}^i - \tilde{p}_j\| \leq \frac{1}{n}\sum_{i' \in [n]} \left\| p_{j+1}^i - p_j^{i'} \right\| \leq \frac{1}{n}\sum_{i \in [n]} r_{\text{up}} = r_{\text{up}}.$$

Finally, from $r \geq r_{\text{lo}}/\sqrt{2}$ and $\lambda \geq 3(r_{\text{up}}/r_{\text{lo}})/\beta$, we can indeed conclude that $\|p_{j+1}^i - \tilde{p}_j\| \leq 0.5\lambda r$ as desired. □

4.3 Aggregation Error Guarantee

By using the guarantee of the bounding circle from the previous section and the concentration of the noise distributions, we can now prove the full utility guarantee as stated below in Theorem 18. Note that Corollary 13 is an immediate consequence of Theorem 18 because it is known that $\sigma(\epsilon_a, \delta_a) \leq O(\sqrt{\log(1/\delta_a)}/\epsilon_a)$ [31].

Theorem 18. *Let*

$$\Lambda := \left(\sqrt{m \log(m/\beta)} \cdot \sigma(\epsilon_a, \delta_a) + \log(1/\beta)/\epsilon_c \right).$$

Suppose that Assumption 12 holds with $\gamma \geq 2^{-k}$. If $n \geq \kappa(\log(1/\beta)/\epsilon_c + \log(k/\beta)/\epsilon_b + (r_{\text{up}}/r_{\text{lo}})^2 \cdot \log(1/\delta_b)/\epsilon_b + \Lambda)$ and $\lambda \geq 3(r_{\text{up}}/r_{\text{lo}})/\beta$, then with probability at least $3/4$, the output $\hat{\mathbf{p}}$ of GaussianLocalBoundingCircle *satisfies*

$$F(\tilde{\mathbf{p}}, \hat{\mathbf{p}}) \leq r_{\text{up}} \cdot O(\Lambda/n), \text{ and}$$
$$F(C, \hat{\mathbf{p}}) \leq r_{\text{up}}\,(1 + O(\Lambda/n)).$$

Proof of Theorem 18. Notice that, from Lemma 15 and the triangle inequality, $F(\tilde{\mathbf{p}}, \hat{\mathbf{p}}) \leq r_{\text{up}} \cdot O(\Lambda/n)$ implies $F(C, \hat{\mathbf{p}}) \leq r_{\text{up}}\,(1 + O(\Lambda/n))$. Thus, it suffices to just show the former.

To do this, observe that, with probability $1-\beta$, we have $|n - \hat{n}| \leq \Delta_n := O(\log(1/\beta)/\epsilon_c)$. Secondly, from standard concentration bounds for the Gaussian distribution, we also have that with probability $1-\beta$, the Gaussian noise random variables sampled on line 12 all have magnitude at most $\Delta_g := O\left(\sqrt{\log(m/\beta)} \cdot \sigma'\right)$. Thirdly, from Lemma 17, with probability $1-5\beta$, we have that

$$p_1^1, \ldots, p_1^n \in B_{\lambda r}(c_0), \tag{1}$$

and

$$\|p_{j+1}^i - \tilde{p}_j\| \leq 0.5\lambda r, \tag{2}$$

for all $i \in [n]$ and $j \in [m-1]$. We will condition on all these events for the rest of the proof. Since the second event holds, we have

$$\left\| \hat{n} \cdot \hat{p}_j' - \sum_{i \in [n]} \left(\Pi(p_j^i) - c_{j-1} \right) \right\| \leq \sqrt{2} \cdot \Delta_g. \tag{3}$$

We will now show by an induction on j that $\|\hat{p}_j - \tilde{p}_j\| \leq \lambda r \cdot O(\Lambda/n)$ for all $j \in [m]$. Notice that Observation 8 then implies that $F(\tilde{\mathbf{p}}, \hat{\mathbf{p}}) \leq \lambda r \cdot O(\Lambda/n)$ as desired.

Base case. From (1), we can conclude that $p_1^i - c_0 = \Pi(p_1^i - c_0)$ for all $i \in [n]$. Thus, from (3) and the definition of $\tilde{p}_1^i$, we get

$$\left\|\hat{n} \cdot \hat{p}_1' - n \cdot (\tilde{p}_1 - c_0)\right\| \leq \sqrt{2} \cdot \Delta_g. \tag{4}$$

As a result, from the triangle inequality, we can derive

$$\begin{aligned}
&\|\hat{p}_1 - \tilde{p}_1\| \\
&\leq \frac{1}{n}\|\hat{n} \cdot (\hat{p}_1 - c_0) - n \cdot (\tilde{p}_1 - c_0)\| \\
&\quad + \frac{1}{n}\|(n - \hat{n}) \cdot (\hat{p}_1 - c_0)\| \\
&= \frac{1}{n}\left\|\hat{n} \cdot \Pi(\hat{p}_1') - n \cdot (\tilde{p}_1 - c_0)\right\| + \frac{1}{n}\|(n - \hat{n}) \cdot \hat{p}_1'\| \\
&\leq \frac{1}{n}\left\|\hat{n} \cdot \hat{p}_1' - n \cdot (\tilde{p}_1 - c_0)\right\| + \frac{1}{n}\|(n - \hat{n}) \cdot \hat{p}_1'\| \\
&\overset{(4)}{\leq} \frac{O(\Delta_g)}{n} + \frac{O(\Delta_n)}{n} \cdot \lambda r \\
&\leq \lambda r \cdot O(\Lambda/n),
\end{aligned}$$

where the second inequality follows from the fact that $\tilde{p}_1 - c_0 \in B_{r'}$ and thus the projection Π does not increase the distance. Thus, for sufficiently large κ, the right hand side is at most $0.5\lambda r$.

Inductive step. Suppose that the statement holds for some $j \in [m-1]$. Observe that the error guarantee $\|\hat{p}_j - \tilde{p}_j\| \leq \lambda r \cdot O(\Lambda/n)$ implies that $\|\hat{p}_j - \tilde{p}_j\| \leq 0.5\lambda r$ when κ is sufficiently large. This, together with (2), ensures that $p_{j+1}^i - c_j = \Pi(p_{j+1}^i - c_j)$. The rest of the proof of the inductive step then follows exactly the same as that of the base case, with $0, 1$ replaced by $j, j+1$ respectively. □

4.4 Robustness of the Algorithm

We conclude by discussing relaxations of Assumption 12 for which a guarantee similar to Corollary 13 still holds. Notice that when averaging each point, we can tolerate an additional error of $c \cdot r_{\text{up}}$ for a sufficiently small constant c while still retaining essentially the same error; notice also that, due to the projection, each point contributes only up to $\lambda \cdot r_{\text{up}}$ to the summation. Similarly, in the bounding box accuracy proof, it suffices if, say, a 0.6 fraction of the points belong to the "correct" box B. This means that we can maintain the guarantee as in Corollary 13 while (i) allowing an $O_\beta(r_{\text{up}}/\lambda)$ fraction of points at each index to be completely arbitrary and (ii) for the remaining points, allowing them to deviate from its corresponding point on the curve by as much as $O(r_{\text{up}})$. This theoretically demonstrates the robustness of our method.

5 Experiments

In this section, we present experimental results evaluating the performance of our algorithm compared to the baselines, and the effect of various parameters.

As stated earlier, we use the Fréchet distance as the dissimilarity metric; we remark that we also experimented with other metrics including the dynamic time warping (DTW) distance and the general trends are similar. We omit these results for brevity.

5.1 Datasets Description

We have conducted experiments on three data sets.

5.1.1 Pigeon Flight Trajectory Data

This dataset (Pigeon) consists of flight paths of 7 pigeons released 24 times from one site and 8 pigeons released 20 times from three other sites each [44]. All four flight paths lead to the same destination. While pigeons have no privacy concerns, we use this dataset as a proxy for real-world trajectories. As we will see, there is a lot of variation in their flight paths (compared to what one would expect in a GPS dataset from public transport), which will test the robustness of our algorithm against data with outliers.

5.1.2 NYC Bus Data

This dataset (NYCBus)[4] contains bus location (latitude and longitude), route name, and timestamp data (among other fields), collected from the NYC MTA SIRI Real Time data feed and the MTA GTFS Schedule data. We use 101 trajectories representing unique routes from the NYC Bus Data in our experiments.

We use each trajectory as a ground truth and generate n different trajectories by sampling m equidistant points from it and shifting them randomly. More precisely, let $\mathbf{p}^* = (p_1^*, \ldots, p_M^*)$ be a trajectory in the NYC Bus Data, where p_i^* consists of the true latitude and longitude of the bus route; this naturally defines a piecewise-linear curve C^*, which we assume to be the ground truth. We define m points $c_1^*, \ldots, c_m^*$ on C^* such

4 Available at https://www.kaggle.com/stoney71/new-york-city-transport-statistics.

that they are placed at equal distances with $c_1^* = p_1^*$ and $c_m^* = p_M^*$. For convenience, let $c_0^* = 2c_1^* - c_2^*$ and $c_{m+1}^* = 2c_m^* - c_{m-1}^*$. We next perturb these points to mimic the behavior of a user traveling on the curve C; this defines a sample trajectory from C^*. To generate the sample, we first uniformly sample x from $[-0.2, 0.2]$. If $x \geq 0$, we define $p_i = c_i^* + x \cdot (c_{i+1}^* - c_i^*)$ for $i \in [m]$; if $x < 0$, we define $p_i = c_i^* + x \cdot (c_i^* - c_{i-1}^*)$. Now, $\mathbf{p} = (p_1, \ldots, p_m)$ is the perturbed trajectory given by our model, generated from the ground truth $\mathbf{p}^*$.

We repeat this process n times for each of the 101 trajectories, and the resulting n trajectories are then used as the input to our algorithms and to the baselines for DP trajectory aggregation.

5.1.3 Character Trajectories Data Set

This dataset (HWChar)[5] consists of around 200 examples each for 20 classes, where each class is a lower case English alphabet written with a single stroke. We assume distance unit in meters for each character sample while calculating Fréchet distance in evaluations.

5.2 Baselines and Implementation Details

In addition to GaussianGlobalBoundingCircle and GaussianLocalBoundingCircle, we also implemented as a baseline the Gaussian mechanism with a "trivial" bounding circle, i.e., for the two GPS-based data sets (Pigeon and NYCBus[6]), adding noise proportional to the smallest circle containing $[-180, 180] \times [-90, 90]$.

We stress that, although the baselines may seem weak, we are not aware of any previous trajectory aggregation algorithm that can be naturally adapted to satisfy DP. The only exception is perhaps the work of He et al. [40] which considers private synthetic trajectories; these synthetic trajectories can then be used to compute a private aggregated trajectory. However, the algorithm from [40] requires much larger datasets (each of their experiments uses at least 4 million trajectories whereas ours uses at most one thousand trajectories).

5.2.1 Pre-Processing

Algorithm 3 assumes that each input trajectory has the same number of points. To satisfy this, we add a preprocessing step where each input trajectory is first resampled to construct an equidistant trajectory (in the same way as we construct an equidistant trajectory for NYCBus). This does not affect the DP guarantees as it is applied to each input trajectory individually.

5.2.2 Post-Processing

For NYC Bus dataset, sometimes the "raw" trajectories output by our algorithm were noisy in a manner uncharacteristic of real-world road networks (caused by the independent noise addition in our algorithm). This motivates a post-processing step to smoothen the raw output trajectory via piecewise-linear regression. Suppose that the raw output trajectory defines a curve C_r. In the *smoothening* step, we find a curve C_s with the minimum number of linear segments such that $F(C_r, C_s) < \theta$ for some threshold θ. This θ is a configurable tolerance on deviation from the output. Due to the post-processing property of DP [31], the smoothened track remains DP. Thus smoothening can yield "realistic-looking" tracks and even slightly reduce the error; unfortunately, we do not have any formal accompanying statement.

Figure 5 shows a comparison between a raw aggregated trajectory and a smoothened one, together with the ground truth, from one of the runs of our GaussianLocalBoundingCircle algorithm. Note that we only apply post-processing on NYC Bus data set but not on the other two, since a priori pigeon flying routes and strokes need not be (close to) piecewise linear.

5.2.3 Parameters

In all of our experiments, we set $\delta = 10^{-4}$ unless stated otherwise. For both the GaussianGlobalBoundingCircle and GaussianLocalBoundingCircle, we let $\epsilon_a = 0.5\epsilon, \epsilon_b = 0.3\epsilon$, and $\epsilon_c = 0.2\epsilon$, while we let $\delta_a = \delta_b = 0.5\delta$.

For the bounding circle computation, we set the maximum depth k to be 16 and the inflation factor λ to be 1.2 throughout, as we observed that this setting results in reasonably good performances across different combinations of n, ϵ, m.

5 Available at https://archive.ics.uci.edu/ml/datasets/Character+Trajectories.

6 GaussianGlobalBoundingCircle for NYCBus is roughly similar to a bounding circle that covers New York City.

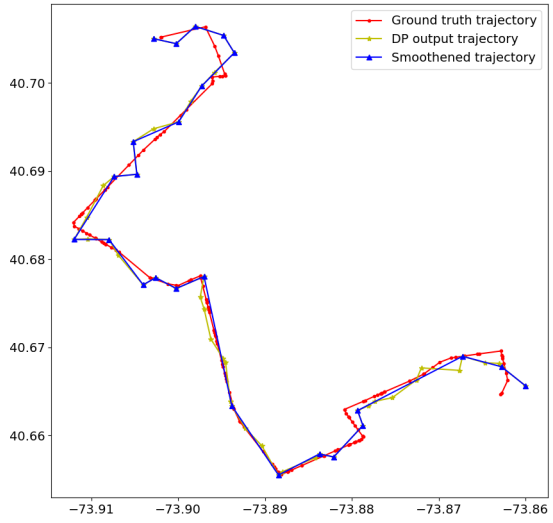

Fig. 5. Illustration of post-processing by smoothening.

5.3 Results

For each data point presented below, we run the algorithms 10 times on the n input trajectories. We then compute the average Fréchet distance of the output and the "true aggregate" over those runs. In the case of NYCBus, the "true aggregate" is the ground truth trajectory used to generate the input trajectories. For Pigeon and HWChar, these empirical datasets have no clear ground truth, and thus we use the average of all trajectories as the "true aggregate".

5.3.1 Comparison Against Baselines

We start with the comparison between GaussianLocalBoundingCircle, GaussianGlobalBoundingCircle, and the "trivial" baseline. We report the average error in the case of $\epsilon = 4, m = 50$ in Table 1. Our GaussianLocalBoundingCircle algorithm clearly demonstrates significant advantage (by an order of magnitude) over the other algorithms. This is indeed the result of having a much smaller noise proportional to just the local bounding circle, instead of the global or trivial bounding circles. "Trivial Box" and "Global Box" baselines perform similarly on HWChar because the latitude, longitude ranges for this dataset are small.

	Pigeon	NYCBus	HWChar
(Baseline) Trivial Box	5,208,500	4,401,392	74.30
(Baseline) Global Box	7,649.10	4,161.31	78.80
Gaussian Local Box	338.93	200.87	44.01

Table 1. Fréchet distance (in meters) of different algorithms.

A visualization of the resulting trajectories obtained by each algorithm is given in Figure 6. Given its clear advantage, we will henceforth just focus on GaussianLocalBoundingCircle.

5.3.2 Effect of Varying Different Parameters

To understand the effect of each parameter on the quality of the algorithm's output, we keep all but one parameter the same and report the average Fréchet distances for different values of that parameter. The parameters of interest here are the privacy parameter ϵ, the number n of users, and the number m of points per trajectory.

In Table 2, we fix n but vary one of ϵ or m. In the former case, as expected, when ϵ increases, the distance decreases, since larger ϵ means adding smaller amounts of noise. Arguably the most interesting behavior is observed in the latter case, i.e., varying m but fixing ϵ. We see a non-monotonic pattern in this case: the distance first decreases as m increases but, after m becomes sufficiently large, the distance starts to increase. This demonstrates the tension between the error from the approximation and the error from Gaussian noise addition. When m is small, most of the error comes from the fact that there are too few points to suffice for a good approximation of the ground truth. On the other hand, a large m means that more noise is added to each point. Such a tension is also corroborated by the utility analysis in Theorem 18 (where r_{up} is the approximation error).

In Table 3, we fix $\epsilon = 4$, $m = 50$, and vary n. Again, as expected, when n increases, the distance decreases. This is simply because the same amount of noise is divided over a larger number of users. This experiment could only be done on NYCBus since we can control the number of generated samples for that dataset.

6 Conclusions

In this work, we proposed an algorithm for privately aggregating trajectories, employing the concept of a local bounding circle. We empirically demonstrated its utility on real-world datasets, showing that it gives highly accurate results even for moderate values of ϵ (e.g., $\epsilon \in [1, 5]$) and number of users (e.g., ≥ 200). It remains interesting to push this further to obtain an algorithm for smaller number of users, say, less than 50. Another

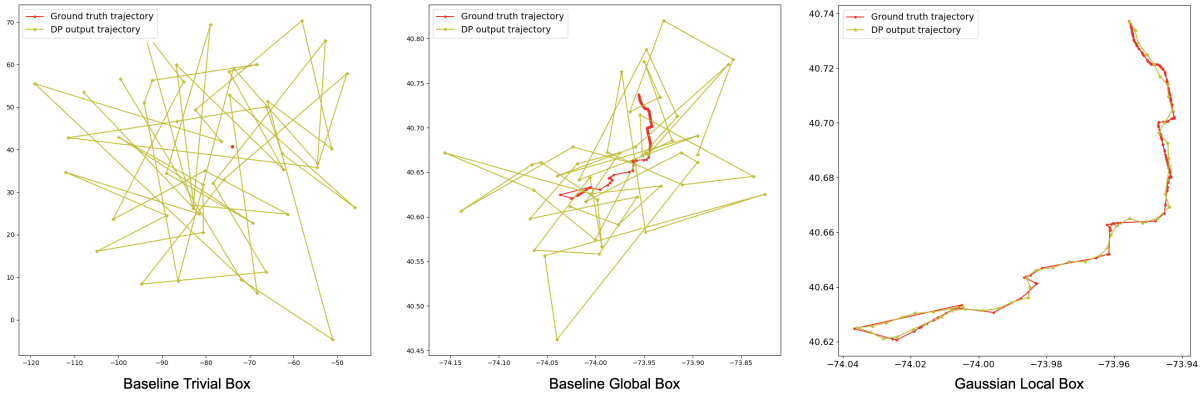

Fig. 6. Sample illustration of different algorithms for a trajectory from NYCBus.

Pigeon

$\epsilon(m = 50)$	2.0	2.4	3.2	4.0	5.0
Distance	539.20	457.70	405.07	338.93	311.60
$m(\epsilon = 4)$	10	20	35	50	80
Distance	380.67	326.48	328.22	338.93	384.89

NYCBus

$\epsilon(m = 50)$	1.0	1.2	2.0	2.4	3.2	4.0
Distance	367.8	315.7	249.3	237.9	228.6	222.9
$m(\epsilon = 1)$	35	50	80	120	160	200
Distance	457.5	403.7	335.1	308.2	421.4	466.4

HWChar

$\epsilon(m = 50)$	2.0	2.4	3.2	4.0	5.0
Distance	76.16	66.34	51.09	44.01	36.17
$m(\epsilon = 4)$	5	10	20	35	50
Distance	23.41	15.32	21.13	33.16	43.47

Table 2. Fréchet distance (in meters) vs ϵ and m for the three datasets.

n	150	200	500	1000
Distance	398.79	309.79	207.70	190.53

Table 3. Fréchet distance (in meters) vs number of user trajectories for NYCBus ($\epsilon = 4$ and $m = 50$).

interesting research direction is to algorithmically exploit the underlying geometry (e.g., road networks) better.

Under simple and natural assumptions on the trajectories and the underlying ground truth curve, we proved that the private aggregated trajectory output by our algorithm is close, in Fréchet distance, to the ground truth. As stated earlier, a different way to formulate the trajectory aggregation problem is via clustering-like objectives (see, e.g., [14, 15, 27]). An interesting direction here would be to obtain DP algorithms with provably guarantees for such objectives as well.

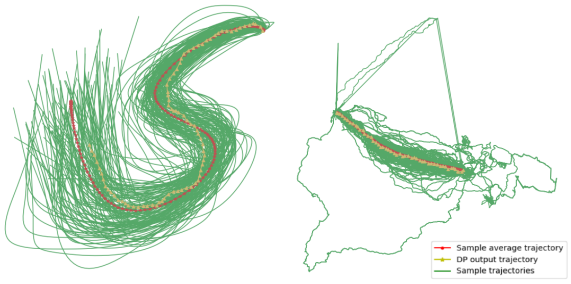

Fig. 7. Illustration of non-private and DP aggregated trajectory for 's' in HWChar (left) and 'Horspath' trajectory in Pigeon (right).

Finally, there are relaxations of standard DP notions that are more tailored towards location-based data, such as *geo-indistinguishibility* [5]. It is interesting to explore whether such notions can allow better utility; this might also require generalizing these DP variants to accommodate a sequence of points (rather than just a single point), which itself might be of independent interest.

Acknowledgements

We are grateful to Dean Scarff for introducing us to the problem and for many insightful discussions.

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] Opinion 05/2014 on anonymisation techniques, 2014. Article 29 Data Protection Working Party.

[2] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *CCS*, pages 308–318, 2016.

[3] J. M. Abowd. The US Census Bureau adopts differential privacy. In *KDD*, pages 2867–2867, 2018.

[4] P. K. Agarwal, M. de Berg, J. Gao, L. J. Guibas, and S. Har-Peled. Staying in the middle: Exact and approximate medians in $\mathbb{R}_1$ and $\mathbb{R}_2$ for moving points. In *CCCG*, pages 43–46, 2005.

[5] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *CCS*, pages 901–914, 2013.

[6] Apple Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 2017.

[7] M. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang. Differentially private clustering in high-dimensional Euclidean spaces. In *ICML*, pages 322–331, 2017.

[8] J. Baldus and K. Bringmann. A fast implementation of near neighbors queries for Fréchet distance (GIS Cup). In *SIGSPATIAL*, pages 1–4, 2017.

[9] B. Balle and Y. Wang. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *ICML*, pages 403–412, 2018.

[10] A. Beimel, K. Nissim, and U. Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *RANDOM*, pages 363–378, 2013.

[11] L. Bonomi. Mining frequent patterns with differential privacy. *PVLDB*, 6(12):1422–1427, 2013.

[12] L. Bonomi and L. Xiong. A two-phase algorithm for mining sequential patterns with differential privacy. In *CIKM*, pages 269–278, 2013.

[13] M. Brankovic, K. Buchin, K. Klaren, A. Nusser, A. Popov, and S. Wong. (k, l)-medians clustering of trajectories using continuous dynamic time warping. In *SIGSPATIAL*, pages 99–110, 2020.

[14] K. Buchin, A. Driemel, J. Gudmundsson, M. Horton, I. Kostitsyna, M. Löffler, and M. Struijs. Approximating (k, ℓ)-center clustering for curves. In *SODA*, pages 2922–2938, 2019.

[15] K. Buchin, A. Driemel, and M. Struijs. On the hardness of computing an average curve. In *SWAT*, pages 19:1–19:19, 2020.

[16] K. Buchin, A. Driemel, N. van de L'Isle, and A. Nusser. klcluster: Center-based clustering of trajectories. In *SIGSPATIAL*, pages 496–499, 2019.

[17] M. Bun, K. Nissim, and U. Stemmer. Simultaneous private learning of multiple concepts. *JMLR*, 20:94–1, 2019.

[18] M. Bun and T. Steinke. Average-case averages: Private algorithms for smooth sensitivity and mean estimation. In *NeurIPS*, pages 181–191, 2019.

[19] A. Chang, B. Ghazi, R. Kumar, and P. Manurangsi. Locally private k-means in one round. In *ICML*, 2021.

[20] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi. Broadening the scope of differential privacy using metrics. In *PoPETS*, pages 82–102, 2013.

[21] R. Chen, G. Ács, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In T. Yu, G. Danezis, and V. D. Gligor, editors, *CCS*, pages 638–649, 2012.

[22] A. Cohen and K. Nissim. Towards formalizing the GDPR's notion of singling out. *PNAS*, 117(15), 2020.

[23] T. Cunningham, G. Cormode, H. Ferhatosmanoglu, and D. Srivastava. Real-world trajectory sharing with local differential privacy. *VLDB*, 14(11):2283–2295, 2021.

[24] D. Desfontaines, J. Voss, B. Gipson, and C. Mandayam. Differentially private partition selection. *PoPETS*, 2022(1):339–352, 2022.

[25] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *NeurIPS*, pages 3571–3580, 2017.

[26] J. Dong, A. Roth, and W. J. Su. Gaussian differential privacy. *CoRR*, abs/1905.02383, 2019.

[27] A. Driemel, A. Krivosija, and C. Sohler. Clustering time series under the Fréchet distance. In *SODA*, pages 766–785, 2016.

[28] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

[29] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[30] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

[31] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[32] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.

[33] L. Fan and L. Xiong. Real-time aggregate monitoring with differential privacy. In *CIKM*, pages 2169–2173, 2012.

[34] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *KDD*, pages 63–72, 1999.

[35] S. Ghane, L. Kulik, and K. Ramamohanarao. TGM: A generative mechanism for publishing trajectories with differential privacy. *IEEE Internet Things J.*, 7(4):2611–2621, 2020.

[36] B. Ghazi, R. Kumar, and P. Manurangsi. Differentially private clustering: Tight approximation ratios. In *NeurIPS*, 2020.

[37] S. Gopi, P. Gulhane, J. Kulkarni, J. H. Shen, M. Shokouhi, and S. Yekhanin. Differentially private set union. In *ICML*, pages 3627–3636, 2020.

[38] A. Greenberg. Apple's "differential privacy" is about collecting your data – but not your data. *Wired, June*, 13, 2016.

[39] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70, 2010.

[40] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava. DPT: differentially private trajectory synthesis using hierarchical reference systems. *PVLDB*, 8(11):1154–1165, 2015.

[41] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *PVLDB*, 7(12):1155–1166, 2014.

[42] C. Li, B. Palanisamy, and J. Joshi. Differentially private trajectory analysis for points-of-interest recommendation. In *BigData Congress*, pages 49–56, 2017.
[43] M. Lyu, D. Su, and N. Li. Understanding the sparse vector technique for differential privacy. *PVLDB*, 10(6):637–648, 2017.
[44] R. Mann, R. Freeman, O. Osborne, R. Garnett, C. Armstrong, J. Meade, D. Biro, T. Guilford, and S. Roberts. Objectively identifying landmark use and predicting flight trajectories of the homing pigeon using Gaussian processes. *The Royal Society Interface*, 2010.
[45] C. Meehan and K. Chaudhuri. Location trace privacy under conditional priors. In *AISTATS*, pages 2881–2889, 2021.
[46] N. Meratnia and R. A. de By. Aggregation and comparison of trajectories. In *SIGSPATIAL*, pages 49–54, 2002.
[47] K. Nissim, A. Bembenek, A. Wood, M. Bun, M. Gaboardi, U. Gasser, D. R. O'Brien, and S. Vadhan. Bridging the gap between computer science and legal approaches to privacy. *Harvard Journal of Law & Technology*, 31:687–780, 2016 2018.
[48] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.
[49] K. Nissim and U. Stemmer. Clustering algorithms for the centralized and local models. In *ALT*, pages 619–653, 2018.
[50] K. Nissim, U. Stemmer, and S. P. Vadhan. Locating a small cluster privately. In *PODS*, pages 413–427, 2016.
[51] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *STOC*, pages 765–774, 2010.
[52] S.Durocher and D.Kirkpatrick. The projection median of a set of points. *CG*, 42(5):364–375, 2009.
[53] S. Shankland. How Google tricks itself to protect Chrome user privacy. *CNET, October*, 2014.
[54] D. M. Sommer, S. Meiser, and E. Mohammadi. Privacy loss classes: The central limit theorem in differential privacy. *PoPETS*, 2019(2):245–269, 2019.
[55] U. Stemmer and H. Kaplan. Differentially private k-means with constant multiplicative error. In *NeurIPS*, pages 5436–5446, 2018.
[56] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal*, 29(1):3–32, 2020.
[57] M. van Kreveld, M. Löffler, and F. Staals. Central trajectories. *J. Computational Geometry*, 8(1), 2017.

A Proof of Theorem 3

In this section, we provide a proof of Theorem 3; we stress that this is not a novel contribution of the paper and we only provide the proof for completeness. There are multiple ways to prove this result. Here we use the notion of *Gaussian Differential Privacy (GDP)* [26] for convenience. Although the original definition of GDP is in terms of f-DP which is in turn defined based on a hypothesis testing scenario, it can also be equivalently defined as follows [26, Corollary 2.13].

Definition 19 (Gaussian Differential Privacy [26])**.** *For any $\mu > 0$, a mechanism $\mathcal{M}$ is μ-GDP iff, for all $\epsilon \geq 0$, $\mathcal{M}$ is $(\epsilon, \delta(\epsilon))$-DP where*

$$\delta(\epsilon) = \Phi\left(-\frac{\epsilon}{\mu} + \frac{\mu}{2}\right) - e^{\epsilon}\Phi\left(-\frac{\epsilon}{\mu} - \frac{\mu}{2}\right).$$

A key property of GDP that we will use is the following composition theorem, which works even in the case of adaptive composition [26, Corollary 3.3].

Theorem 20 (Adaptive Composition for GDP [26])**.** *Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$, where $\mathcal{M}_i : X \times Y_1 \times \cdots \times Y_{i-1}$, be mechanisms such that $\mathcal{M}_i(\cdot, y_1, \ldots, y_{i-1})$ is μ_i-GDP for all $i = 1, \ldots, k$. Then, the composed mechanism $\mathcal{M}(x) := (y_1, \ldots, y_k)$ defined by $y_i = \mathcal{M}_i(x, y_1, \ldots, y_{i-1})$ is $\sqrt{\mu_1^2 + \cdots + \mu_k^2}$-GDP.*

We will also use is [9, Theorem 8], which provides the tight (ϵ, δ)-DP guarantee for the Gaussian mechanism.

Theorem 21 (Gaussian Mechanism DP Guarantee [9])**.** *For any $\epsilon, \delta \geq 0$, the Gaussian mechanism (defined in Section 2.1.1) satisfies (ϵ, δ)-DP iff*

$$\delta \geq \Phi\left(-\frac{\epsilon\sigma}{\Delta_2 f} + \frac{\Delta_2 f}{2\sigma}\right) - e^{\epsilon}\Phi\left(-\frac{\epsilon\sigma}{\Delta_2 f} - \frac{\Delta_2 f}{2\sigma}\right).$$

We are now ready to prove Theorem 3.

Proof of Theorem 3. Let $\sigma' = \Delta\sqrt{m} \cdot \sigma(\epsilon, \delta)$.

From Theorem 21, each f_i satisfies $(\epsilon', \delta(\epsilon'))$-DP for all $\epsilon' \geq 0$ and

$$\begin{aligned}\delta(\epsilon') &= \Phi\left(-\frac{\epsilon'\sigma'}{\Delta_2 f_i} + \frac{\Delta_2 f_i}{2\sigma'}\right) - e^{\epsilon'}\Phi\left(-\frac{\epsilon'\sigma'}{\Delta_2 f_i} - \frac{\Delta_2 f_i}{2\sigma'}\right) \\ &\leq \Phi\left(-\frac{\epsilon'\sigma'}{\Delta} + \frac{\Delta}{2\sigma'}\right) - e^{\epsilon'}\Phi\left(-\frac{\epsilon'\sigma'}{\Delta} - \frac{\Delta}{2\sigma'}\right),\end{aligned}$$

where the inequality follows from $\Delta_2 f_i \leq \Delta$.

From Definition 19, this means that f_i is (Δ/σ')-GDP. Then, Theorem 20 ensures that the m-fold (possible adaptive) composition of the Gaussian mechanism applied on $f_1, \ldots, f_m$ satisfies $(\sqrt{m}\Delta/\sigma')$-GDP. Thus, from Definition 19, this implies that the mechanism is (ϵ, δ^*)-DP for

$$\begin{aligned}\delta^* &= \Phi\left(-\frac{\epsilon\sigma'}{\sqrt{m}\Delta} + \frac{\sqrt{m}\Delta}{2\sigma'}\right) - e^{\epsilon}\Phi\left(-\frac{\epsilon\sigma'}{\sqrt{m}\Delta} - \frac{\sqrt{m}\Delta}{2\sigma'}\right) \\ &= \Phi\left(-\epsilon\sigma(\epsilon, \delta) + \frac{1}{2\sigma(\epsilon, \delta)}\right)\end{aligned}$$

$$- e^{\epsilon}\Phi\left(-\epsilon\sigma(\epsilon,\delta) - \frac{1}{2\sigma(\epsilon,\delta)}\right)$$
$$\leq \delta,$$

where the inequality follows from our choice of $\sigma(\epsilon,\delta)$. Therefore, we can conclude that the algorithm is (ϵ,δ)-DP as desired. □

B Proof of Theorem 7

In this section, we prove Theorem 7. We in fact prove a more general version where a user can contribute an arbitrary vector of non-negative real numbers (not necessarily integers), as long as its ℓ_1-norm is bounded. Theorem 7 then corresponds to the special case where the ℓ_1-norm is $\Delta = L$ and where each user contribution is integral.

B.1 Laplace Mechanism with Thresholding

In the below, we assume that there are n users where each user i holds a vector $\mathbf{x}^i = (x^i_1, \ldots, x^i_B) \in \mathbb{R}^B_{\geq 0}$ such that $\|\mathbf{x}^i\|_1 \leq \Delta$ and the goal is to output an estimate of $\sum_{i\in[n]} \mathbf{x}^i$.

The truncated Laplace distribution, denoted by $\mathrm{tLap}_{b,t}$, is the distribution supported on $[-t,t]$ where its probability density at x is proportional to $e^{-|x|/b}$ for all $x \in [-t,t]$.

We use notation $\mathrm{Thresh}_\tau : \mathbb{R} \to \mathbb{R} \cup \{\perp\}$ for a real number $\tau \in \mathbb{R}$, to denote the function

$$\mathrm{Thresh}_\tau(x) = \begin{cases} \perp & \text{if } x \leq \tau, \\ x & \text{otherwise.} \end{cases}$$

We also overload the notation Thresh_τ to take in a vector and perform thresholding on each coordinate.

The clipped truncated Laplace mechanism with parameters b, t, τ is an algorithm that outputs $\mathrm{Thresh}_\tau\left(\mathbf{z} + \sum_{i\in[n]} \mathbf{x}^i\right)$ where $\mathbf{z} \in \mathbb{R}^B$ is a noise vector, whose entries are sampled i.i.d. from $\mathrm{tLap}(b,t)$.

We remark that this algorithm is t-accurate since the noise added to each entry has magnitude less than or equal to t.

B.2 DP Analysis

Due to the post-processing property of DP, it suffices to consider the truncated Laplace mechanism that outputs $\mathbf{z} + \sum_{i\in[n]} \mathbf{x}^i$ where $\mathbf{z} \sim \mathrm{tLap}(b,t)$. For this, we prove the following:

Theorem 22. *For any $\epsilon > 0$ and any $\delta \in (0,1)$, the truncated Laplace mechanism with parameter $b \geq \Delta/\epsilon$ and $t \geq b \cdot (\epsilon + \ln(1/\delta))$ is (ϵ,δ)-DP.*

B.2.1 Additional Preliminaries

For any continuous distribution μ, we write f_μ to denote its probability density function.

Definition 23. *For a given mechanism $\mathcal{M}$ and two adjacent databases X, X', its* privacy loss *random variable $PL_{\mathcal{M},X,X'}$ is a random variable selected by first picking $o \sim \mathcal{M}(X)$ and let the privacy loss random be $\ln\left(f_{\mathcal{M}(X)}(o)/f_{\mathcal{M}(X')}(o)\right)$.*

We will use the following well-known result[7].

Lemma 24. *Let $\mathcal{M}$ be any mechanism. Let $\omega_{X,X'}$ denote the privacy loss random variable with respect to $\mathcal{M}, X, X'$. If we have*

$$\Pr[\omega_{X,X'} \geq \epsilon] \leq \delta$$

for all adjacent X, X', then $\mathcal{M}$ is (ϵ,δ)-DP.

B.2.2 Proof of Theorem 22

Proof of Theorem 22. We write $\mathcal{M}$ as a shorthand for the mechanism.

Consider two adjacent databases X, X'. We will give the proof in the case where X' results from removing a user from X; the other case can be proved analogously. Suppose that $X = (\mathbf{x}^1, \ldots, \mathbf{x}^n)$ whereas $X' = (\mathbf{x}^1, \ldots, \mathbf{x}^{n-1})$. From Lemma 24, it suffices for us to show that

$$\Pr_{o\sim\mathcal{M}(X)}\left[\ln\left(\frac{f_{\mathcal{M}(X)}(o)}{f_{\mathcal{M}(X')}(o)}\right) \geq \epsilon\right] \leq \delta. \tag{5}$$

Let us write $\mathbf{y} = \sum_{i\in[n-1]} \mathbf{x}^i$. In our setting, we may write the left hand side as

$$\Pr_{o\sim\mathcal{M}(X)}\left[\ln\left(\frac{f_{\mathcal{M}(X)}(o)}{f_{\mathcal{M}(X')}(o)}\right) \geq \epsilon\right]$$
$$= \Pr_{\mathbf{z}\sim\mathrm{tLap}(b,t)^{\otimes B}}\left[\ln\left(\frac{f_{\mathcal{M}(X)}(\mathbf{y}+\mathbf{x}^n+\mathbf{z})}{f_{\mathcal{M}(X')}(\mathbf{y}+\mathbf{x}^n+\mathbf{z})}\right) \geq \epsilon\right]$$

7 See, e.g., [9, Theorem 5], which has an even tighter version of the bound.

$$= \Pr_{\mathbf{z}\sim \mathrm{tLap}(b,t)^{\otimes B}}\left[\ln\left(\frac{f_{\mathrm{tLap}(b,t)^{\otimes B}}(\mathbf{z})}{f_{\mathrm{tLap}(b,t)^{\otimes B}}(\mathbf{x}^n+\mathbf{z})}\right)\geq\epsilon\right]$$
$$= \Pr_{\mathbf{z}\sim \mathrm{tLap}(b,t)^{\otimes B}}\left[\ln\left(\frac{\prod_{j\in[B]} f_{\mathrm{tLap}(b,t)}(z_i)}{\prod_{j\in[B]} f_{\mathrm{tLap}(b,t)}(x_i^n+z_i)}\right)\geq\epsilon\right]$$
$$= \Pr_{\mathbf{z}\sim \mathrm{tLap}(b,t)^{\otimes B}}\left[\sum_{j\in[B]}\ln\left(\frac{f_{\mathrm{tLap}(b,t)}(z_i)}{f_{\mathrm{tLap}(b,t)}(x_i^n+z_i)}\right)\geq\epsilon\right]. \tag{6}$$

Now, recall that the probability density function $f_{\mathrm{tLap}(b,t)}(a)$ is proportional to $e^{-|a|/b}$ for all $a\in[-t,t]$. This means that, if $z_j\in[-t,t-x_j^n]$ for all $j\in[B]$, then we have

$$\sum_{j\in[B]}\ln(f_{\mathrm{tLap}(b,t)}(z_j)/f_{\mathrm{tLap}(b,t)}(x_j^n+z_j))$$
$$\leq \sum_{j\in[B]} x_j^n/b = \|\mathbf{x}^n\|_1/b\leq\epsilon.$$

Plugging this back into (6), we have

$$\Pr_{o\sim\mathcal{M}(X)}\left[\ln\left(\frac{f_{\mathcal{M}(X)}(o)}{f_{\mathcal{M}(X')}(o)}\right)\geq\epsilon\right]$$
$$\leq \Pr_{\mathbf{z}\sim \mathrm{tLap}(b,t)^{\otimes B}}\left[\bigvee_{j\in[B]} z_j > t-x_j^n\right]$$
$$(*)\leq \sum_{j\in[B]}\Pr_{z_j\sim\mathrm{tLap}(b,t)}[z_j>t-x_j^n]$$
$$\leq \sum_{j\in[B]}\frac{\int_{t-x_j^n}^{t} e^{-|a|/b}da}{\int_{-t}^{t} e^{-|a|/b}da}$$
$$= \frac{1}{2\left(\int_0^t e^{-a/b}da\right)}\left(\sum_{j\in[B]}\int_{t-x_j^n}^{t} e^{-a/b}da\right)$$
$$= \frac{1}{2b\left(1-e^{-t/b}\right)}\left(\sum_{j\in[B]} b\left(e^{-(t-x_j^n)/b}-e^{-t/b}\right)\right)$$
$$= \frac{1}{2\left(e^{t/b}-1\right)}\left(\sum_{j\in[B]}\left(e^{x_j^n/b}-1\right)\right)$$
$$(**)\leq \frac{1}{2\left(e^{t/b}-1\right)}\left(e^{\|\mathbf{x}^n\|_1/b}-1\right)$$
$$\leq \frac{e^{\epsilon}-1}{2\left(e^{t/b}-1\right)}$$
$$\leq \frac{1}{2e^{t/b-\epsilon}}$$
$$\leq \delta,$$

where $(*)$ follows from a union bound and $(**)$ follows from $(e^{u_1}-1)+(e^{u_2}-1)\leq(e^{u_1+u_2}-1)$ for all $u_1,u_2\geq 0$.

Thus, we can conclude that the algorithm is (ϵ,δ)-DP as desired. □